

Silverlight 2 Security

- All code is "sandboxed"
- Sandboxing built on transparency model
 - Transparent: user code (untrusted)
 - Security-critical: platform code (can P/Invoke)
 - Security-safe-critical: platform code (entry points into security-critical code)
- For more information:

<http://blogs.msdn.com/shawnfa/archive/2007/05/09/the-silverlight-security-model.aspx>

Hello, Silverlight 2!




XAML Enhancements

- Silverlight 2 supports a richer dialect of XAML than Silverlight 1.0
 - Rich layout
 - Controls
 - Styles
 - Data binding
 - User controls
- Silverlight XAML is backward-compatible subset of WPF XAML


Controls

Content Controls	Data Controls	Range Controls	Layout Controls	Other Controls
Button	DataGrid	Slider	Canvas	Border
CheckBox	ListBox	Scrollbar	Grid	Calendar
Hyperlink-Button			GridSplitter	DatePicker
Radio-Button			Ink-Presenter	MultiScale-Image
Repeat-Button			StackPanel	TextBox
Scroll-Viewer				Other



Button Control


```
<Button Width="256" Height="128" FontSize="24" Content="Click Me" Click="Button_Click" />
```



```
private void Button_Click(object sender, RoutedEventArgs e)  
{  
    ...  
}
```

Rotated Button

```
<Button Width="256" Height="128" FontSize="24" Content="Click Me" Click="Button_Click">  
    <Button.RenderTransform>  
        <RotateTransform Angle="15" />  
    </Button.RenderTransform>  
</Button>
```



Button with ToolTip

```
<Button Width="256" Height="128" FontSize="24"
Content="Click Me" Click="Button_Click">
  <ToolTipService.ToolTip>
    <StackPanel Background="LightYellow">
      <TextBlock Text="Please click me...pretty please!" />
    </StackPanel>
  </ToolTipService.ToolTip>
</Button>
```



Button with Custom Content

```
<Button Width="256" Height="128" Click="Button_Click">
  <Button.Content>
    <StackPanel Orientation="Horizontal" HorizontalAlignment="Center">
      <Ellipse Width="75" Height="75" Margin="10">
        <Ellipse.Fill>
          <RadialGradientBrush GradientOrigin="0.25,0.25">
            <GradientStop Offset="0.25" Color="White" />
            <GradientStop Offset="1.0" Color="Red" />
          </RadialGradientBrush>
        </Ellipse.Fill>
      </Ellipse>
      <TextBlock FontSize="24" Text="Click Me" VerticalAlignment="Center" />
    </StackPanel>
  </Button.Content>
</Button>
```



Control Templates

```
<Button>
  <Button.Template>
    <ControlTemplate>
      <Grid>
        <Ellipse Width="256" Height="128">
          <Ellipse.Fill>
            <RadialGradientBrush GradientOrigin="0.25,0.25">
              <GradientStop Offset="0.25" Color="White" />
              <GradientStop Offset="1.0" Color="Red" />
            </RadialGradientBrush>
          </Ellipse.Fill>
        </Ellipse>
        <TextBlock FontSize="24" Text="Click Me"
          HorizontalAlignment="Center" VerticalAlignment="Center" />
      </Grid>
    </ControlTemplate>
  </Button.Template>
</Button>
```



TemplateBinding

```
<Button Width="256" Height="128" FontSize="24" Content="Click Me">
<Button.Template>
<ControlTemplate>
<Grid>
<Ellipse Width="{TemplateBinding Width}"
Height="{TemplateBinding Height}">
<Ellipse.Fill>
<RadialGradientBrush GradientOrigin="0.25,0.25">
<GradientStop Offset="0.25" Color="White" />
<GradientStop Offset="1.0" Color="Red" />
</RadialGradientBrush>
</Ellipse.Fill>
</Ellipse>
<TextBlock FontSize="{TemplateBinding FontSize}" Text="Click Me"
HorizontalAlignment="Center" VerticalAlignment="Center" />
</Grid>
</ControlTemplate>
</Button.Template>
</Button>
```

Styles

- Styles provide level of indirection between visual properties and their values
 - Define style as XAML resource
 - Apply style using {StaticResource} markup extension
- Can be scoped globally or locally
- Combine styles and templates to “stylize” controls with custom visual trees

Defining and Using Styles

```
Style name           Prevents style from being
                    applied to non-Buttons
↓                   ↓
<Style x:Key="ButtonStyle" TargetType="Button">
<Setter Property="Width" Value="256" />
<Setter Property="Height" Value="128" />
<Setter Property="FontSize" Value="24" />
</Style>
...
<Button Style="{StaticResource ButtonStyle}" ... />
<Button Style="{StaticResource ButtonStyle}" ... />
<Button Style="{StaticResource ButtonStyle}" ... />
<Button Style="{StaticResource ButtonStyle}" Width="128" ... />
```

Explicit property value overrides style property value

Scoping Styles

App.xaml

```
<!-- Global style -->  
<Application.Resources>  
  <Style x:Key="ButtonStyle" TargetType="Button">  
    ...  
  </Style>  
</Application.Resources>
```

Page.xaml (or other XAML file)

```
<!-- Local style -->  
<UserControl.Resources>  
  <Style x:Key="ButtonStyle" TargetType="Button">  
    ...  
  </Style>  
</UserControl.Resources>
```

Combining Styles and Templates

```
<Style x:Key="EllipticalButton" TargetType="Button">  
  <Setter Property="Template">  
    <ControlTemplate TargetType="Button">  
      <Grid>  
        <Ellipse Width="{TemplateBinding Width}" Height="{TemplateBinding Height}">  
          <Ellipse.Fill>  
            <RadialGradientBrush GradientOrigin="0.25,0.25">  
              <GradientStop Offset="0.25" Color="White" />  
              <GradientStop Offset="1.0" Color="Red" />  
            </RadialGradientBrush>  
          </Ellipse.Fill>  
        </Ellipse>  
        <ContentPresenter Content="{TemplateBinding Content}"  
          HorizontalAlignment="Center" VerticalAlignment="Center"  
          FontSize="{TemplateBinding FontSize}" />  
      </Grid>  
    </ControlTemplate>  
  </Setter>  
</Style>  
  ...  
<Button Style="{StaticResource EllipticalButton}" Content="Click Me"  
  Width="256" Height="128" FontSize="24" Click="Button_Click" />
```

Styles

Demo

Data Binding

- Permits properties of one object to be bound to properties of another
 - Target property must be DependencyProperty
 - Source property can be any type of property
 - Source can be a collection if target supports binding to collections
- {Binding} markup extension provides declarative support for data binding

Simple Data Binding

```
<TextBox x:Name="Input" />  
<TextBlock x:Name="Output" Text="{Binding Input.Text}" />
```

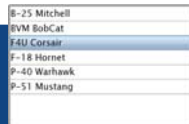
Get Text property of TextBlock from Text property of source

ListBox Data Binding

```
<ListBox x:Name="AircraftList" />
```

```
string[] aircraft = {  
    "B-25 Mitchell", "BVM BobCat", "F4U Corsair",  
    "F-18 Hornet", "P-40 Warhawk", "P-51 Mustang"  
};
```

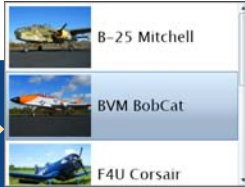
```
AircraftList.ItemsSource = aircraft;
```



Templated Data-Bound ListBox

```
<ListBox x:Name="AircraftList">  
<ListBox.ItemTemplate>  
  <DataTemplate>  
    <StackPanel Orientation="Horizontal">  
      <Image Source="{Binding Thumbnail}" Margin="5" />  
      <TextBlock Text="{Binding Name}" Margin="5" FontSize="24"  
        HorizontalAlignment="Center" VerticalAlignment="Center" />  
    </StackPanel>  
  </DataTemplate>  
</ListBox.ItemTemplate>  
</ListBox>
```

AircraftList.ItemsSource = aircraft;



Data Binding

Demo

DOM Integration

- System.Windows.Browser namespace contains classes for accessing browser DOM
 - HtmlPage, HtmlWindow, and others
- Managed -> unmanaged
 - Access DOM from managed code
 - Call JavaScript functions from managed code
- Unmanaged -> managed
 - Call managed code from JavaScript
 - Process DOM events with managed code

Alerting the User

```
HtmlPage.Window.Alert ("Error!");
```

Calling JavaScript Functions from C#

C#

```
HtmlPage.Window.Invoke("changeColor", "Blue");
```

JavaScript

```
function changeColor(args)
{
    var color = args; // color == 'Blue'
    ...
}
```

Exposing C# Methods to JavaScript

Allows class to support scriptable members

Name used to refer to this object in JavaScript

```
[ScriptableType]
public partial class Page : UserControl
{
    public Page()
    {
        ...
        HtmlPage.RegisterScriptableObject("magic", this);
    }

    [ScriptableMember]
    public void ChangeColor(string color) { ... }
}
```

Script-callable method

Calling C# Methods from JavaScript

```
var control = document.getElementById('SilverlightControl');  
control.content.magic.ChangeColor('Red');
```

ID of Silverlight control

Registered name *Method name*

DOM Integration



File I/O

- General-purpose file I/O not permitted
- OpenFileDialog can be used to open files
 - User interaction constitutes permission needed to safely open files
 - No SaveFileDialog in Beta 1
- Isolated storage can be used to persist data locally subject to quotas

OpenFileDialog

```
// Display an image selected by the user
OpenFileDialog ofd = new OpenFileDialog();
ofd.Filter =
  "JPEG Files (*.jpg;*.jpeg)|*.jpg;*.jpeg|All Files (*.*)|*.*";
ofd.FilterIndex = 1;

if ((bool)ofd.ShowDialog())
{
  using (Stream stream = ofd.SelectedFile.OpenRead())
  {
    BitmapImage bi = new BitmapImage();
    bi.SetSource(stream);
    MyImage.Source = bi; // MyImage is a XAML Image object
  }
}
```

Isolated Storage

- System.IO.IsolatedStorage contains classes for safe, secure, per-user local storage
 - Ideal for personalization settings
 - Virtualized (physical location hidden)
- Scope can be per-app or per-site, but...
- Storage per-user is max 1 MB per domain
 - IsolatedStorageFile.IncreaseQuotaTo method increases quota if user approves

Writing to Isolated Storage

```
using (IsolatedStorageFile store =
  IsolatedStorageFile.GetUserStoreForApplication())
{
  using (IsolatedStorageFileStream stream =
    new IsolatedStorageFileStream("Settings.xml",
      FileMode.Create, store))
  {
    using (StreamWriter writer = new StreamWriter(stream))
    {
      // TODO: Write to stream with StreamWriter
    }
  }
}
```

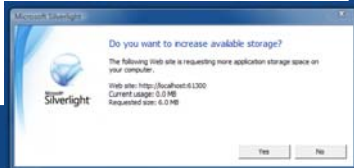
Reading from Isolated Storage

Caution: Throws *IsolatedStorageException* if file doesn't exist

```
using (IsolatedStorageFile store =  
    IsolatedStorageFile.GetUserStoreForApplication())  
{  
    using (IsolatedStorageFileStream stream =  
        new IsolatedStorageFileStream("Settings.xml",  
            FileMode.Open, store))  
    {  
        using (StreamReader reader = new StreamReader(stream))  
        {  
            // TODO: Read from stream with StreamReader  
        }  
    }  
}
```

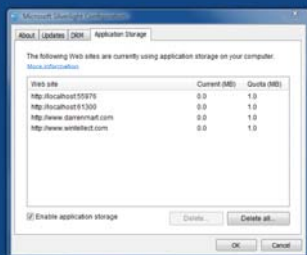
Requesting a Quota Increase

```
Int64 needed = 1048576;  
Int64 available = store.AvailableFreeSpace;  
  
if (available < needed)  
{  
    if (store.IncreaseQuotaTo(store.Quota + needed))  
    {  
        // Granted  
    }  
    else  
    {  
        // Denied  
    }  
}
```



Managing Isolated Storage

- Silverlight.Configuration.exe



Isolated Storage



Networking

- Silverlight 2 has rich networking support
 - SOAP/XML Web services via WCF proxies
 - HTTP services (POX, REST, RSS, ATOM) via `HttpWebRequest` and `WebClient` classes
 - Socket support, asset downloads over HTTP, syndication classes, and more
- Cross-domain access supported using Flash-compatible or Silverlight XML policy files

Cross-Domain Network Calls

- Allowed if target domain has XML policy file in place permitting calls from other domains
 - `Crossdomain.xml` – Requires `domain="*"` allowing calls from any domain
 - `Clientaccesspolicy.xml` – Can restrict access to certain domains only
- Policy file must be located at domain root

[http://msdn2.microsoft.com/en-us/library/cc197955\(VS.95\).aspx](http://msdn2.microsoft.com/en-us/library/cc197955(VS.95).aspx)

Sample Policy File

```
<?xml version="1.0"?>  
<!DOCTYPE cross-domain-policy SYSTEM  
  "http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">  
<cross-domain-policy>  
  <allow-access-from domain="*" />  
</cross-domain-policy>
```

Allow access from all domains

HttpRequest

- Delegate-based HTTP networking API
 - Supports asynchronous operation only
- Provides control over wire format
 - GET/POST/PUT/DELETE (REST)
 - Customization of HTTP headers
- Completion methods called on background threads

Calling a REST Service

```
HttpRequest request = (HttpRequest) HttpRequest.Create  
  (new Uri("http://contoso.com/weather/98052"));  
request.BeginGetResponse  
  (new AsyncCallback(OnGetResponseCompleted), request);  
...  
private void OnGetResponseCompleted(IAsyncResult ar)  
{  
  HttpRequest request = (HttpRequest)ar.AsyncState;  
  HttpResponse response =  
    (HttpResponse)request.EndGetResponse(ar);  
  
  using (StreamReader reader =  
    new StreamReader(response.GetResponseStream()))  
  {  
    string result = reader.ReadToEnd();  
    ...  
  }  
}
```

WebClient

- Event-based HTTP networking API
 - Commonly used to download assets
 - DownloadStringAsync - String
 - OpenReadAsync – Stream (binary)
 - Can also be used to perform uploads
- Fires progress and completion events and supports cancellation of pending requests
 - Event handlers execute on UI thread

Downloading and Consuming RSS

```
WebClient wc = new WebClient();
wc.OpenReadCompleted +=
    new OpenReadCompletedEventHandler(OnDownloadCompleted);
wc.OpenReadAsync(new Uri
    ("http://feeds.feedburner.com/AbcNews_TopStories"));
...
void OnDownloadCompleted(object sender,
    OpenReadCompletedEventArgs e)
{
    using (XmlReader reader = XmlReader.Create(e.Result))
    {
        SyndicationFeed feed = SyndicationFeed.Load(reader);
        RssLB.ItemsSource = feed.Items; // Bind to ListBox
    }
}
```

Downloading Binary Assets

```
WebClient wc = new WebClient();
wc.DownloadProgressChanged += new
    DownloadProgressChangedEventArgs(OnProgressChanged);
wc.OpenReadCompleted += new
    OpenReadCompletedEventHandler(OnDownloadCompleted);
wc.OpenReadAsync(new Uri("Assets/Flight.wmv", UriKind.Relative));
...
void OnDownloadCompleted(object sender,
    OpenReadCompletedEventArgs e)
{
    Stream result = e.Result;
    // TODO: Use result
}
```

Networking



Discussion